ภาษา LOGØ
อินเตอร์เพรตเตอร์

บุญเลิศ เอี่ยมทัศนา

โปรแกรมที่เป็นตัวเอกของบทความชุดนี้คือ โปรแกรมที่อยู่ในยูนิต INTERPRT ทั้งหมดนี้จะร่วมกันทำหน้าที่เป็นโปรแกรมตัวแปลภาษาในแบบอินเตอร์เพรตเตอร์ ผมจึงเรียกว่าอินเตอร์เพรตเตอร์ ซึ่งผมเขียนขึ้นตามแนวความคิดของผมเท่านั้น ไม่ถือว่าเป็นหลักการหรือหลักวิชาแต่อย่างใด

โปรแกรมหลักของล็อกศูนย์

เพื่อเป็นการทบทวนและต้องนำไปใช้ด้วย จึงขอนำโปรแกรมหลักของภาษาล็อกศูนย์มากล่าวถึง ดังแสดงในรูป M2 โปรแกรมนี้เดิมผมเก็บไว้ในไฟล์ชื่อ MAIN2.PAS เพื่อการพัฒนาต่อไป แต่ตอนนี้ผมตัดสินใจยุติไว้เพียงนี้จึงเปลี่ยนชื่อไฟล์เป็น LOGØ.PAS สำหรับการพัฒนาต่อไปทางด้านกราฟิกให้ใช้งานได้จริงจังจะเปลี่ยนชื่อใหม่ แต่ยังไม่ทราบเลยครับว่าจะใช้ชื่ออะไรดี

การทำงานของโปรแกรมหลักเริ่มจากให้ยูนิต LOGØ เตรียมค่าต่าง ๆ ให้ (ซึ่งผมพึ่งมาเห็นในตอนนี้เองว่าเป็นวิธีเขียนโปรแกรมที่ไม่ถูกต้อง เพราะผู้อ่านโปรแกรมหลักจะไม่ทราบเลยว่าได้มีการเตรียมค่าต่าง ๆ แล้ว) เสร็จแล้วให้ทำงานที่หน้าต่างล่าง แล้วเข้าวงรอบ REPEAT

ในวงรอบ REPEAT เริ่มด้วยการเตรียมค่าที่ต้องเตรียมทุกครั้ง ยกเลิกค่าคีย์ที่ค้างอยู่ในคีย์บอร์ดบัฟเฟอร์ แล้วให้ผู้ใช้ป้อนคำสั่งซึ่งผู้ใช้จะป้อนเพียงคำสั่งเดียวหรือหลายคำสั่งก็ได้ ดังตัวอย่างคำสั่งในรูปที่ 1 แต่จะนำคำแรกมาตรวจดู โดยที่

- ถ้าเป็นสตริงว่างหมายถึงผู้ใช้

ต้องการขึ้นบรรทัดใหม่

- ถ้าเป็น TOO คือ 'to' หมายถึงต้องการเรียกเอดิเตอร์ (ซึ่งกรณีนี้ยังไม่ได้ดำเนินการ)

- ถ้าเป็น QUIT คือ 'quit' หมายถึงต้องการออกจากภาษาล็อกศูนย์

- ถ้าเป็นคำที่เกี่ยวกับคอนโซล ก็ให้ทำคำนั้น ๆ ซึ่งอยู่ในยูนิต CONSOLE

- คำที่เหลือถือว่าเป็นคำสั่งเต่า จึงให้ดำเนินการด้วย DoTurtleComm ในยูนิต INTERPRT ดังจะกล่าวในฉบับนี้

หลังจากดำเนินการแล้วจะให้ปิดท้ายข้อความที่ผู้ใช้ป้อนด้วย > แล้วดำเนินการเกี่ยวกับข้อผิดพลาด จะทำซ้ำอยู่เช่นนี้จนกว่าผู้ใช้ป้อนคำว่า quit

ยูนิต INTERPRT

ยูนิตนี้จะดำเนินการทั้งหลายทั้งปวงในการแปลโปรแกรมเต่าให้เครื่องปฏิบัติ มีโพรซีเดอร์ที่ให้โปรแกรมหลักเรียกใช้ได้เพียงหนึ่งเดียวคือ DoTurtleComm ดังแสดงส่วนหัวไว้ในรูป I1 และมีโปรแกรมย่อยที่จะดำเนินการต่าง ๆ อีก คือ

DoTurtleComm

เมื่อผู้ใช้ป้อนคำสั่งนั้นอาจมีหลาย ๆ คำสั่งได้ดังกล่าวแล้ว เทียบได้กับโปรแกรมย่อยหนึ่งแต่เป็นโปรแกรมย่อยที่ไม่มีส่วนหัว จึงให้ DoTurtleComm ต่อท้ายด้วย END; ให้ เพื่อให้เหมือนกับโปรแกรมย่อยอื่น ๆ แล้วเรียกใช้ TurtleComm เพื่อดำเนินการต่อไป ดังแสดงในรูป I8

TurtleComm

ความจริงแล้ว TurtleComm ก็คือ DoTurtleComm ตัวจริงในแง่ของการตั้งชื่อ เพราะเป็นโปรแกรมที่จะดำเนินการในการแปลโปรแกรมเต่าอย่างแท้จริง มีส่วนหัวอยู่ในรูป I2 แต่ตัวโปรแกรมหลักอยู่ในรูป I8

ก่อนที่ผมจะกล่าวต่อไปขอให้ดูโปรแกรมหลักในรูป M2 จะเห็นว่าเมื่อเรียกใช้ DoTurtleComm ได้ผ่านค่าที่จะให้แก่ V1 และ V2 เป็น 1 นั่นคือทุกครั้งที่ผู้ใช้สั่งคำสั่งใหม่ จะยกเลิกตัวแปรที่เป็นโลคอลเดิมทั้งหมด

อีกเรื่องหนึ่งคือ px เมื่อเริ่มต้นจะเป็น 0 คือชี้ไปที่ต้นอะเรย์ที่ผู้ใช้ป้อน ส่วนค่าใน wrd คือคำแรกในอะเรย์นี้ และ cx จะชี้อยู่หลังอักขระตัวสุดท้ายใน wrd ทั้งนี้ก็เพื่อให้เป็นที่สังเกตไว้ครับ

การแปลของ TurtleComm จะดำเนินการแปลโปรแกรมเต่าครั้งละ 1 ถ้อยแถลง โดยที่ถ้อยแถลงในภาษาล็อกศูนย์มีอยู่ 3 แบบ คือเป็น ถ้อยแถลงคำสั่ง ถ้อยแถลงให้ค่า และถ้อยแถลงโครงสร้างซึ่งแบ่งย่อยเป็น IF กับ

REPEAT เท่านั้น จึงดำเนินการโดยตรวจคำแรก ถ้าเป็น

- make จะเป็นถ้อยแถลงให้ค่า

- repeat จะเป็นถ้อยแถลง REPEAT

- if จะเป็นถ้อยแถลง IF

- ที่เหลือให้เป็นถ้อยแถลงคำสั่ง

เมื่อเป็นถ้อยแถลงใดให้แยกไปปฏิบัติตามถ้อยแถลงนั้น ๆ ซึ่งถ้อยแถลงคำสั่งอาจมีถ้อยแถลงทั้ง 4 นี้ ซ้อน ๆ กันต่อไปอีกก็ได้ จึงเหมาะสำหรับการเขียนโปรแกรมในแบบเรียกตนเอง คือถ้าไม่มีที่ผิดจะเรียกตนเองจนกว่าจะพบคำว่า end; หมายถึงจบโปรแกรมย่อยเต่า หรือ stop ซึ่งหมายถึงให้ยุติการเรียกตนเองของโปรแกรมย่อยเต่า

แต่การยุติการเรียกตนเองจริง ๆ ของล็อกศูนย์จะเกิดขึ้นต่อเมื่อ ได้มีการส่งกลับเรียกตนเองใหม่ส่งกลับอีก จนเป็น end; ของ px = 0 ดังจะกล่าวต่อไป

DoProcedure

เมื่อเป็นถ้อยแถลงคำสั่งจะมาทำที่โปรแกรมย่อยนี้ เริ่มด้วยการตรวจว่ามีการกดคีย์หรือไม่ ถ้ามีจะให้ err = 100 เพื่อยุติการรันโปรแกรมเต่า (เช่น เมื่อติดอยู่ในวงรอบของการเรียกตนเองในโปรแกรมเต่า) ดังแสดงในรูป I4

แล้วนำคำมาตรวจ ถ้าเป็นคำมาตรฐานก็ให้ดำเนินการตามที่กล่าวในยูนิต LIBRARY (ไมโครฯ 50) ถ้าเป็นคำใหม่ก็จะดำเนินการตามที่จะกล่าวต่อไป แต่ขอให้สังเกตว่าในการตรวจว่าเป็นคำใหม่หรือไม่ ได้ผ่านค่า np กับ nc ซึ่งหมายถึง โพรซีเดอร์-ที่ กับอักขระ-ที่ ซึ่งเป็นตัวแปรตัวใหม่ไปให้ UserDefine

UserDefine

จะตรวจว่าคำที่ส่งมาเป็นคำใหม่หรือไม่ โดยเทียบกับคำแรกของโพรซีเดอร์ต่าง ๆ ตั้งแต่โพรซีเดอร์ที่ 1 ถึง pm-1 (ดู DoLoad ในรูป C2 ในไมโครฯ 50)

ในการเทียบจะเทียบอักขระแรกก่อน หากตรงกันจะเทียบอักขระตัวต่อ ๆ ไปด้วย Equal หากตรงกันทั้งหมดจะออกจากโพรซีเดอร์นี้ได้โพรซีเดอร์-ที่ ใน px และอักขระ-ที่ ใน cx ผ่านกลับไปให้ np และ nc ตามลำดับ และได้ UserDefine เป็นจริง

หากไม่ตรงกันจะตรวจในโพรซีเดอร์ต่อ ๆ ไปด้วยการเพิ่มค่า px จนกว่า px = pm แสดงว่าไม่มีคำนี้ในฐานข้อมูล ให้ UserDefine เป็นเท็จกลับไป ดังแสดงในรูป I2

DoUser

หากเป็นคำใหม่ DoProcedure จะเรียกใช้ DoUser โดยผ่านลำดับที่ของโพรซีเดอร์เดิมมาใน px และของ

M2

```
{$M  65520,65536,655360}
{$V-}
PROGRAM  Log0;
uses  crt,Log00,console,cal,interprt;

VAR      wrd : String;  px,cx : Word;
         def  : ConT;  ch : Char;

(*-----------------      Other  Unit   ------------------*)

procedure  DoEdit(px : Word);
  begin  end;

(*.....................................................*)

BEGIN  { MAIN }
  OpenWindow(user);

  REPEAT
    err := 0;  perr := 0;  cerr := 0;  px := 0;
    Write('<');
    while  KeyPressed  do  ch := ReadKey;
    ReadStream(px);  cx := 1;
    GetToken(px,cx,wrd);
    IF  wrd = ''  THEN  WriteLn('>')
    ELSE IF  wrd = TOO  THEN  DoEdit(px)
    ELSE IF  wrd = QUIT  THEN  fin := TRUE
    ELSE IF  ConsComm(wrd,def)  THEN  DoConsole(def,px,cx)
    ELSE  DoTurtleComm(wrd,px,cx,1,1);

    IF  wrd <> ''  THEN  Write('> ');
    IF  (err <> 0) AND (err <> 100)  THEN
       WriteLn('ERROR = ',err);
  UNTIL  fin;

Write('clear screen (y/n) ');  ch := Upcase(ReadKey);
if  ch = 'Y' then  begin  Window(1,1,80,25);  ClrScr end;
END.
```

โพรซีเดอร์ใหม่มาใน np กับอักขระ-ที่ของเดิมมาใน cx ของใหม่มาใน nc ดังแสดงในรูป I3

การทำงานของ DoUser เริ่มด้วยกำหนดช่องในอะเรย์ที่จะเก็บตัวแปรของโพรซีเดอร์ใหม่ คือที่ให้ค่า nv1 และ nv2 เป็น v2 จากนั้นจะดำเนินการเกี่ยวกับการผ่านค่า เช่น ส่วนหัวของโพรซีเดอร์ใหม่กำหนดไว้เป็น box:side และโพรซีเดอร์เก่าเรียกใช้เป็น box 10 ก็จะเป็นการนำค่า 10 มาให้แก่ side

แต่ส่วนหัวของโพรซีเดอร์อาจจะไม่มีอาร์กิวเมนต์เลย หรือมีหนึ่ง (เช่น box) หรือมีมากกว่าหนึ่ง จึงให้ดำเนินการโดย (ยกตัวอย่าง box ดังกล่าวมา)

- อ่านคำที่อยู่ต่อจาก box ซึ่งจะต้องเป็นตัวแปร

- นำมาตรวจในวงรอบ WHILE ว่าเป็นตัวแปรหรือไม่ ถ้าใช่จึงจะทำงานในวงรอบ WHILE คือ

- อ่านค่า 10 จากโพรซีเดอร์เดิมให้แก่ฟิลด์ข้อมูล แต่ถ้าไม่ใช่ค่าจะออกจากโปรแกรมนี้ และยุติการรันเนื่องจากข้อผิดพลาดที่เกิดขึ้นใน CanGetValue

- เมื่อถูกต้องให้นำชื่อตัวแปรคือ side มาเก็บในฟิลด์ชื่อ แล้วเพิ่มค่า nv2 ขึ้นหนึ่ง การดำเนินการเช่นนี้เท่ากับเป็นการกำหนดตัวแปรโลคอลของโพรซีเดอร์ใหม่ 1 ตัว

- หลังจากนี้ให้อ่านคำต่อไปในโพรซีเดอร์ box ถ้าเป็นตัวแปรก็จะทำงานในวงรอบ WHILE อีก

เมื่อผ่านวงรอบ WHILE มาแล้ว คำที่อยู่ใน wrd คือคำที่เป็นถ้อยแถลงของโพรซีเดอร์ box จึงให้เรียกใช้ TurtleComm นั่นคือ เป็นการเรียกตนเองของ TurtleComm ด้วย เพราะ DoUser เป็นโปรแกรมย่อยภายในของ TurtleComm การที่ให้เรียก ณ จุดนี้ เพราะต้องการเปลี่ยนตัวแปร px, cx, v1 และ v2 เป็น np, nc, nv1 และ nv2 ตามลำดับ

การดำเนินการเช่นนี้จะทำให้ TurtleComm ย้ายไปทำงานในส่วนถ้อยแถลงของโพรซีเดอร์ใหม่ไปจนถึง END; ของโพรซีเดอร์ใหม่แล้ว จึงจะกลับมาที่โพรซีเดอร์เดิม ทำงานไปจนถึง END; ของโพรซีเดอร์เดิม และถ้าในโพรซีเดอร์ใหม่มีการเรียกใช้โพรซีเดอร์ใหม่อีก ก็จะไปทำงานในโพรซีเดอร์หลังนี้อีกกี่ชั้นก็ได้ (จนสแต็กเต็ม)

DoAssignment

เมื่อเป็นถ้อยแถลงให้ค่าคือขึ้นต้นด้วยคำว่า make โพรซีเดอร์ TurtleComm จะแยกให้มาทำงานที่โพรซีเดอร์ DoAssignment การให้ค่าในภาษาล็อกศูนย์ดำเนินการเป็น 2 แบบ คือ ให้ค่าพร้อมทั้งกำหนดตัวแปรเป็นโกลบอล กับให้ค่าแก่ตัวแปรเดิมไม่ว่าจะเป็นโกลบอลหรือโลคอล การทำงานของโปรแกรมย่อยนี้จึงเป็นดังนี้

- อ่านคำที่อยู่ถัดจาก make ซึ่งจะต้องขึ้นต้นด้วยเครื่องหมายคำพูดคู่ (เครื่องหมาย " หมายถึงชื่อตัวแปร เครื่องหมาย : หมายถึงค่าของตัวแปร) แล้วหาค่าซึ่งเป็นคำถัดไปอีกมาเก็บในตัวแปร dat
- ตรวจว่าเป็นตัวแปรโลคอลหรือไม่ ถ้าใช่ให้ค่าใน dat แก่ตัวแปรนี้
- ถ้าไม่ใช่ตัวแปรโลคอล ให้ตรวจว่าเป็นตัวแปรโกลบอลหรือไม่ ถ้าใช่ให้ค่าใน dat แก่ตัวแปรนี้
- ถ้าไม่ใช่ตัวแปรโกลบอลเดิม ให้กำหนดตัวแปรโกลบอลใหม่แล้วให้ค่าใน dat (ในโปรแกรมให้ค่าก่อนกำหนดชื่อ)

โปรแกรมในท่อนหลังเป็นการเลื่อน cx ไปไว้ท้ายข้อมูลที่เป็นสตริงหรือลิสต์ ซึ่งค่าใน fch ตอนนี้จะเปลี่ยนเป็นเครื่องหมาย " หรือ [ด้วย CanGetValue ถ้าข้อมูลเป็นสตริงหรือลิสต์ ดังแสดงในรูป I5

DoRepeat

เมื่อพบคำว่า repeat โพรซีเดอร์ TurtleComm จะเรียกใช้ DoRepeat ซึ่งวงรอบ repeat ของภาษาล็อกศูนย์จะเทียบเท่ากับวงรอบ FOR ในภาษาปาสคาล ตัวเลขที่ตามมา (ขอให้ดูรูปที่ 1) คือ จำนวนครั้งที่ให้ทำงานในวงรอบนี้ สำหรับถ้อยแถลงที่ให้ทำซ้ำจะอยู่ในวงเล็บ [ ] การทำงานของโปรแกรมย่อยนี้ดังแสดงในรูป I6 เป็นดังนี้

- อ่านค่าซึ่งจะเป็นจำนวนที่ให้ทำซ้ำมาเก็บใน dat แล้วแปลงเป็นเลขจำนวนเต็มให้แก่ num
- เก็บค่า cx ไว้ใน cc เพื่อเริ่มที่ถ้อยแถลงแรกทุกครั้งที่เริ่มวงรอบใหม่ ดังที่เห็นให้ค่าแก่ cx ที่ต้นวงรอบ FOR
- ในวงรอบ FOR มีวงรอบ REPEAT อยู่ ซึ่งจะอ่านคำมาดำเนินการด้วย DoProcedure ทั้งนี้เพราะในวงรอบ repeat ของภาษาล็อกศูนย์ จะมีแต่ถ้อยแถลงคำสั่งเท่านั้น การดำเนินการเช่นนี้ไม่ใช่การเรียกตนเองทำให้ทำงานได้เร็วขึ้น
- จะทำงานซ้ำอยู่ในวงรอบ REPEAT จนพบเครื่องหมาย] แสดงว่าหมดถ้อยแถลงที่ให้ทำซ้ำ กลับไปเริ่มต้นวงรอบ FOR ใหม่ จนกว่าจะครบจำนวนครั้งที่กำหนด
- แต่ถ้ามีข้อผิดพลาดจะออกจากโปรแกรมนี้

DoIf

เมื่อพบคำว่า if โพรซีเดอร์ TurtleComm จะเรียกใช้ DoIf ซึ่งถ้อยแถลง if ของภาษาล็อกศูนย์จะประกอบด้วยคำว่า if เงื่อนไข และถ้อยแถลงที่ต้องการให้ปฏิบัติอยู่ภายในวงเล็บก้ามปู (ขอให้ดูในรูปที่ 1) และไม่มี else การดำเนินการของโปรแกรมนี้ ดังแสดงในรูป I7 เป็นดังนี้

- อ่านค่าที่เป็นตัวตั้งมาเก็บใน dat1
- อ่านเครื่องหมายมาเก็บใน sign แล้วตรวจว่าเป็นตัวดำเนินการที่ถูกต้องหรือไม่ด้วย IsOperator ถ้าถูกต้องจะได้ลำดับที่ของตัวดำเนินการใน oper
- อ่านค่าที่เป็นตัวกระทำมาเก็บใน dat2
- ดำเนินการเปรียบเทียบตามตัวดำเนินการด้วย CASE ซึ่งถ้าเงื่อนไขเป็นจริงจะได้ว่า cond เป็นจริง
- หาก cond เป็นเท็จจะไม่ปฏิบัติถ้อยแถลงที่ตามมา ซึ่งทำได้โดยเลื่อน cx ไปไว้หลังเครื่องหมาย] ในทางตรงข้ามเมื่อ cond เป็นจริงไม่ดำเนินการใดจะทำให้ TurtleComm ทำถ้อยแถลงที่ตามมา

เรื่องราวของอินเตอร์เพรตเตอร์ซึ่งเป็นพระเอกของบทความชุดนี้มีเพียงเท่านี้ครับ ได้แสดงตัวอย่างโปรแกรมในภาษาล็อกศูนย์ในรูปที่ 2 และผมได้รวบรวมคำสั่งที่จะใช้ได้ไว้ในรูปที่ 3

I1

```
{$V-}
UNIT  InterPrt;

Interface
uses  crt,Log00,console,cal,library;

PROCEDURE  DoTurtleComm(var wrd : String; px,cx,v1,v2 : Word);

(* ---------------------------------------------------------*)
```

I2

```
Implementation

PROCEDURE  TurtleComm(var wrd : String; px : Word; var cx : Word;
                                                  v1,v2 : Word);

FUNCTION  UserDefine(var wrd : String; var px,cx : Word) : Boolean;
VAR    w : Word;

  function  Equal(var wrd1 : String; var wrd2 : StreamT) : Boolean;
  begin
    w := 1;  cx := 1;  Equal := FALSE;
    while  wrd1[w] <> SPACE  do
      if  wrd1[w] <> wrd2[cx]  then  Exit
      else  begin  inc(w);  inc(cx)  end;
    Equal := TRUE;
  end;

BEGIN
  px := 1;  UserDefine := TRUE;
  WHILE  (px < pm)  DO
   IF  (wrd[1] <> proc[px].pntr^[1])  THEN  inc(px)
   ELSE IF  Equal(wrd,proc[px].pntr^)  THEN  Exit
   ELSE  inc(px);
  UserDefine := FALSE;
END;
```

I3

```
PROCEDURE  DoUser(var px,cx : Word; np,nc,v1,v2 : Word);
VAR    nv1,nv2,cc : Word;  dat : DataT;
       ref : Boolean;
BEGIN
  nv1 := v2;  nv2 := v2;  cc := cx;
  GetToken(np,nc,wrd);
  WHILE  fch = VARSIGN  DO
    BEGIN
    if  not CanGetValue(px,cx,v1,v2,vars[nv2].data,ref)  then  Exit;
    vars[nv2].name := wrd;  inc(nv2);
    GetToken(np,nc,wrd);
    END;

  IF  (err = 0)  THEN   TurtleComm(wrd,np,nc,nv1,nv2);
END;
```

I4

```
PROCEDURE  DoProcedure(var wrd : String;
                       var px,cx : Word; v1,v2 : Word);
VAR  idn : ComT; np,nc : Word;  wrdz : String;
BEGIN
  if  KeyPressed  then  err := 100;
  wrdz := wrd + SPACE;
  IF  StandardIden(wrd,idn)  THEN
    DoStandard(idn,px,cx,v1,v2)
  ELSE IF UserDefine(wrdz,np,nc)  THEN
    DoUser(px,cx,np,nc,v1,v2)
  ELSE err := 1;
END;
```

I5

```
PROCEDURE  DoAssignment(var px,cx : Word; v1,v2 : Word);
VAR     dat : DataT;  vx,gx : Word;
        endstrm : Char; fine,ref : Boolean;
BEGIN  { tem }
  GetToken(px,cx,wrd);
  IF  (fch = QUOT) AND  CanGetValue(px,cx,v1,v2,dat,ref)  THEN
    BEGIN
    while  (wrd <> vars[v1].name) and (v1 <> v2)  do  inc(v1);
    IF  v1 < v2  then  vars[v1].data := dat
    ELSE
      BEGIN
      gx := 1;
      while  (wrd <> gvar[gx].name) and (gx <> gm)  do  inc(gx);
      gvar[gx].data := dat;
      if  gx = gm  then  begin  gvar[gm].name := wrd;  inc(gm) end;
      END;

    if  fch = QUOT  then  endstrm := SPACE
    else if  fch = OPNBLK  then
      begin  endstrm := CLSBLK;  fch := QUOT  end;

    if  fch = QUOT  then
      begin  while  proc[px].pntr^[cx] <> endstrm  do  inc(cx);
             inc(cx)   end;
    END;
END;
```

I6

```
PROCEDURE  DoRepeat(var px,cx : Word; v1,v2 : Word);
VAR     dat : DataT;  c,cc : Word;
        i,num : Integer;  ref : Boolean;
BEGIN
  IF  CanGetValue(px,cx,v1,v2,dat,ref)  THEN
    BEGIN
    cc := cx;  num := Trunc(dat.num);
    FOR  i := 1 TO num  DO
      BEGIN
      cx := cc;
      REPEAT
        GetToken(px,cx,wrd);
        DoProcedure(wrd,px,cx,v1,v2);
      UNTIL  (lch = ']') OR (err <> 0);
      END;
    END
  ELSE  err := 104;
END;
```

I7

```
PROCEDURE  DoIf(var px,cx : Word; v1,v2 : Word);
TYPE    BoolOprT = (eq,gt,lt,ge,le,ne,ns);
CONST   BoolSign    : ARRAY[BoolOprT] OF String[2] =
                    ('=','>','<','>=','<=','<>','');
VAR     n1,n2 : Real;  cond,ref : Boolean;
        dat1,dat2 : DataT;
        wrd,sign : IdenT;  oper : BoolOprT;
        cc : Word;
```

```
FUNCTION  IsOperator(var sgn : IdenT;
                     var op : BoolOprT) : Boolean;
BEGIN
  op := eq;
  while  (sgn <> BoolSign[op]) and (op <> ns)  do
    op := Succ(op);
  IsOperator := op <> ns;
END;


BEGIN  { DoIf }
  IF  CanGetValue(px,cx,v1,v2,dat1,ref)  THEN
    BEGIN
    GetToken(px,cx,sign);
    IF  IsOperator(sign,oper) AND
        CanGetValue(px,cx,v1,v2,dat2,ref)  THEN
      BEGIN
      n1 := Trunc(dat1.num);
      n2 := Trunc(dat2.num);
      CASE  oper  OF
        eq : cond := (n1 = n2);
        gt : cond := (n1 > n2);
        lt : cond := (n1 < n2);
        ge : cond := (n1 >= n2);
        le : cond := (n1 <= n2);
        ne : cond := (n1 <> n2);
      END   { case }
      END   { if }
    ELSE  MarkErr(px,cx,105);

    IF  cond = FALSE THEN
      BEGIN
      WHILE  proc[px].pntr^[cx] <> CLSBLK  DO
        inc(cx);
      inc(cx);
      END;
    END;
END;   { DoIf }
```

```
BEGIN  { TurtleComm }
  IF  (wrd = ENDP) or (wrd = STOP)  THEN  begin  end
  ELSE
    BEGIN
    IF  wrd = MAKE  THEN  DoAssignment(px,cx,v1,v2)
    ELSE IF  wrd = REPEATT  THEN  DoRepeat(px,cx,v1,v2)
    ELSE IF  wrd = IFF  THEN  DoIf(px,cx,v1,v2)
    ELSE  DoProcedure(wrd,px,cx,v1,v2);

    IF  (err = 0) THEN
      BEGIN
      GetToken(px,cx,wrd);
      TurtleComm(wrd,px,cx,v1,v2);
      END;
    END;
END;


PROCEDURE  DoTurtleComm(var wrd : String;
                        px,cx,v1,v2 : Word);
VAR      cc,p    : Word;
```

I8

```
BEGIN
  cc := cx;  p := 1;
  while  proc[px].pntr^[cc] <> #0  do  inc(cc);
  for  p := 1 to NUMCHAR  do
    begin  proc[px].pntr^[cc] := PlsEnd[p];
           inc(cc)  end;
  TurtleComm(wrd,px,cx,v1,v2);
END;


END.
```

```
pr "Hello
pr [ World of Computer]
fd 10  rt 90  fd 10  rt 90  fd 10

make "num  200
pr :num
make  "num  (:num / 10)

if  :num = 20  [fd 10  rt 90  nop]
repeat  :num  [fd 5  rt 1]
repeat  5  [pr "LOGO  nl nop]
```

รูปที่ 1 รูปแบบของคำสั่งเต่าที่สั่งที่คอนโซลได้ คือในการสั่งแต่ละครั้ง (ก่อนกดคีย์ Enter) จะสั่งกี่คำสั่งก็ได้ แต่ในรูปนี้บรรทัดที่ 3 เท่านั้นที่มีหลายคำสั่ง นอกนั้นมีเพียง 1 คำสั่ง คือ 1 ถ้อยแถลง เพื่อแสดงรูปแบบของถ้อยแถลงต่าง ๆ ในภาษาล็อกศูนย์ คือ กลุ่มแรกเป็นถ้อยแถลงคำสั่ง กลุ่มที่สองที่ขึ้นต้นด้วย make เป็นถ้อยแถลงให้ค่า กลุ่มที่สามเป็นถ้อยแถลงโครงสร้าง IF และ repeat ถ้อยแถลงละ 1 บรรทัด (เว้นบรรทัดที่ 3)

```
dobox  fd 10  rt 90  fd 10  rt 90  fd 10
       rt 90  fd 10  rt 90  end;

doboxn :side  repeat 4 [fd :side  rt 90]  end;

doboxes :num :side
  repeat :num [doboxn :side  rt 30]
end;

dostar :side
  repeat 5 [fd :side  rt 144]
end;

docircle
  fd 1  rt 10
  if heading = 0 [stop]
  docircle
end;
```

รูปที่ 2 ตัวอย่างโปรแกรมในภาษาล็อกศูนย์ (ซึ่งเก็บอยู่ในไฟล์ TURTLE.LG0)

รูปที่ 3 คำในภาษาล็อกศูนย์

คำในภาษาล็อกศูนย์

คำสงวน	
make "n v	ให้ค่า v แก่ n หรือกำหนดตัวแปรโกลบอล n แล้วให้ค่า v
if cond [ ]	หาก cond เป็นจริงให้ทำคำสั่งในเครื่องหมายวงเล็บก้ามปู
repeat n [ ]	ให้ทำคำสั่งในวงเล็บก้ามปูซ้ำ n ครั้ง
end;	จบโปรแกรมย่อย
stop	ยุติการทำงานในโปรแกรมย่อยนั้น

→

โพรซีเดอร์

pu	ยกปากกา คือ ไม่ทิ้งรอยทางเดิน
pd	วางปากกา คือ ทิ้งรอยทางเดิน
draw	ให้แสดงภาพเต่าในแบบไม่กระโดด
nodraw	แสดงภาพเต่าในแบบกระโดด
fd n	เดินหน้า n ก้าว
fk n	ถอยหลัง n ก้าว
rt d	หันขวา d ดีกรี
lt d	หันซ้าย d ดีกรี
nop	ไม่มีการปฏิบัติ
cs	ลบจอภาพ แล้วให้เต่าอยู่ตรงกลางจอภาพ หันหัวไปทางทิศ 000
pr	นำข้อมูลที่ตามมาไปออกจอภาพ
n1	ให้ขึ้นบรรทัดใหม่

ฟังก์ชัน

heading	ให้ทิศหัวของเต่าขณะนั้น
Xcor	ให้ตำบลที่ของเต่าในแกน x
Ycor	ให้ตำบลที่ของเต่าในแกน y
rc	ให้ค่าอักขระ 1 ตัวซึ่งอ่านจากคีย์บอร์ด

คำสั่งที่คอนโซล

to	เรียกใช้เอดิเตอร์ (ยังไม่มี)
load [finame]	โหลดไฟล์ finame หากระบุ ถ้าไม่ระบุจะโหลดไฟล์ TURTLE.LG0 ลงหน่วยความจำ
list	แสดงโปรแกรมย่อยให้หน่วยความจำ
save [finame]	เก็บข้อมูลในหน่วยความจำลงไฟล์ finame หากระบุ ถ้าไม่ระบุจะเก็บลงไฟล์ TURTLE.LG0 (ยังไม่มี)
cls	ลบภาพบนจอภาพ
park	ให้เต่าอยู่กลางจอภาพ หันหัวไปทางทิศ 000
new	เริ่มต้นใหม่ (ยกเลิกโปรแกรมและข้อมูลในหน่วยความจำ)

■

กุรุป้อนโปรแกรมตัวอย่างหรือโปรแกรมในภาษาลือกศูนย์อื่น หากคุณคอมพายล์ไฟล์ LOGØ.PAS (MAIN2.PAS เดิม) เป็น LOGØ.EXE ให้ใช้ไซด์คิกในการป้อน แต่ถ้ารันในเทอร์โบควรใช้เอดิเตอร์ของเทอร์โบในการป้อน (หากใช้ไซด์คิกจะมีปัญหาสำหรับเทอร์โบปาสคาล 5.0 เนื่องจากหน่วยความจำไม่พอ)

สำหรับข้อบกพร่องอื่นที่ยังไม่ได้กล่าวและยังไม่ได้แก้ คือ

- การที่ผมกำหนดตัวแปร lch ไว้เป็นโกลบอล ทำให้มีปัญหาในวงรอบ repeat ดังที่ต้องนำคำสั่ง nop มาคั่นก่อนที่จะปิดด้วย]

- ในการเรียกตนเองยังเก็บข้อมูลลงสแต็กมากเกินความจำเป็น ทำให้เรียกตนเองมากครั้งไม่ได้ ดังเช่นในโพรซีเดอร์ docircle (รูปที่ 2) หากเปลี่ยนคำสั่ง rt 10 เป็น rt 1 สแต็กจะเต็ม

ข้อบกพร่องผิดพลาดอื่นก็คงจะมีอีกครับ เพียงแต่ผมไม่ได้ทดลองโปรแกรมโดยละเอียด เพราะไปทำงานอื่น คือผมตั้งใจจะเขียนเอดิเตอร์ในกราฟิกโหมด ดังที่ปูทางไว้แล้วด้วยเรื่อง "ภาษาไทยในกราฟิกโหมด" เสร็จแล้ว ผมจึงจะย้อนกลับมาปรับปรุงโปรแกรมนี้เป็นตัวแปลภาษา LOGX (LOGIC eXercise)

สำหรับโปรแกรมของผมนี้ คุณจะนำไปปรับปรุงเปลี่ยนแปลงอย่างไรก็ได้ ทำให้สมบูรณ์ขึ้นเป็นสิทธิของคุณ (อ้างถึงกันบ้างก็จะขอบพระคุณครับ) เพื่อให้สามารถเขียนโปรแกรมอย่างภาษาโลโกด้วยภาษาไทยได้